sustainability

Article

Design and Implementation of Thermal Comfort System based on Tasks Allocation Mechanism in Smart Homes

Imran Imran [1], Shabir Ahmad [1] and DoHyeun Kim [1,*]

[1] Department of Computer Engineering, Jeju National University, Jeju 63243, Korea; imranjofficial@jejunu.ac.kr (I.); shabir@jejunu.ac.kr (S.A.)

* Correspondence: kimdh@jejunu.ac.kr

Received: 4 September 2019; Accepted: 14 October 2019 ; Published: 21 October 2019

Abstract: The recent trend in the Internet of Things (IoT) is bringing innovations in almost every field of science. IoT is mainly focused on the connectivity of things via the Internet. IoT's integration tools are developed based on the Do It Yourself (DIY) approach, as the general public lacks technical skills. This paper presents a thermal comfort system based on tasks allocation mechanism in smart homes. This paper designs and implements the tasks allocation mechanism based on virtual objects composition for IoT applications. We provide user-friendly drag and drops panels for the new IoT users to visualize both task composition and device virtualization. This paper also designs tasks generation from microservices, tasks mapping, task scheduling, and tasks allocation for thermal comfort applications in smart home. Microservices are functional units of services in an IoT environment. Physical devices are registered, and their corresponding virtual objects are initialized. Tasks are generated from the microservices and connected with the relevant virtual objects. Afterward, they are scheduled and finally allocated on the physical IoT device. The task composition toolbox is deployed on the cloud for users to access the application remotely. The performance of the proposed architecture is evaluated using both real-time and simulated scenarios. Round trip time (RTT), response time, task dropping and latency are used as the performance metrics. Results indicate that even for worst-case scenarios, values of these metrics are negligible, which makes our architecture significant, better and ideal for task allocation in IoT network.

Keywords: Internet of Things; Do-It-Yourself; IoT Applications; Task Level Management; Microservices; Task allocation

---

1. Introduction

Services provided through the world wide web (WWW) serve as a medium for people to create, innovate and share their work with others. The Internet enables the reuse of other's work as a cornerstone for smart and useful things creation. The Internet medium plays a vital role in the realization of IoT [1]. The recent trend in the IoT is bringing innovations in almost every field of life [2]. Hardware boards are electronic modules; IoT users can combine these modules to develop their custom IoT applications based on smarter things. Some of the most popular boards are Adafruit Flora [3], Arduino [4], Intel Edison [5], and Raspberry Pi [6]. Most of the manufacturers of these boards provide API and coding documentation for the programming languages supported by its environment; many of these boards support programming languages such as Python and Java. Users with the necessary skills of its supported languages can start creating IoT applications by customizing the existing code for these boards.

In the IoT network sensors, actuators are the most used vital nodes; other important key nodes are smart objects, RFID tags, and servers. These nodes can perform a minimal amount of tasks as they are different from each other in capabilities such as power consumption, residual energy, available memory, and processing capacity [7]. In order to perform some given IoT application tasks, these nodes must have capability of reconfiguring themselves and interoperate autonomously. Powerful IoT applications require different IoT nodes to collaborate for performing a specific task; all of these tasks performed by a node or more than one nodes brings to the most diverse applications. Examples include IoT applications in smart house, Agriculture and Healthcare. Furthermore, technologies trends showings that in future, more complex applications will be developed by integrating IoT and Artificial intelligence, and other fields' knowledge together. e.g., Machine to Machine (M2M) communication, i.e., smart cars and smart vehicles sharing critical information such as traffic congestion and accident locations to ambulances.

Human psychology has various drivers toward the DIY approach. Usability and Innovations of today's DIY architectures enable new users to do creative things in a simple, easy way. In the IoT field, one of the motivations for applying DIY approaches is the advancement in development Kits such as Arduino, Grove* and Raspberry Pi. Today's IoT research is mainly focused on technologies and research methodologies for enabling the connectivity of things to the Internet intelligently and efficiently. These innovations and research in IoT, particularly in DIY architectures making the general public involved in the extension of these technologies to other fields of science. Therefore, DIY architecture and toolkits are essential for future IoT advancements. New users cannot understand many important contents easily if architecture is not following usability and other ease of use matrices. Hence, one of the goals of this research is to provide a solution to the task allocation problem using a DIY approach focused on user usability and reliability factors.

In this paper, we propose a tasks allocation mechanism for thermal comfort in smart homes. The process is based on task-based managemnt that includes from microservice analysis to task allocation on physical devices. Microservices are used for the generation of Tasks in the IoT environment. These generated tasks are directly associated with IoT resources such as sensors or actuators. A task is associated with a device commonly known as task mapping [8]. Task scheduling is changing the order of these pairs such that tasks that need to be executed first are executed first and vice versa. Task allocation is the process of assigning a task to a physical device and allocating a time slot to the task. Task deployment is the execution of tasks on the physical device at the allocated time. The IoT orchestration architecture we propose provides effective task allocation on physical IoT devices full filing basic requirements of the user and avoid the excess use of constrained resources.

The rest of the paper is structured as follows. Section 2 presents the related work, highlighting the contribution to IoT integration architectures. In Section 3, the proposed methods for Task and Virtual Objects Composition are discussed; Section 4 explains the design of the proposed architecture; Implementation and results are discussed in Section 5. Performance analysis and significance of the proposed architecture is discussed in Section 6; Section 7 concludes the research study and discusses future work direction.

2. Related Work

The IoT vision is the connectivity of smart objects that can be deployed in any space to provide services to people. In order for good user experience, users difficulties in the use of IoT tools must be addressed.Some of the research focuses on the DIY approach for the development of user-friendly IoT platforms; users learn visual programming-based techniques to develop their custom IoT applications.

Philips company is a brand of electronics, famous in the market for electronics products primarily known for manufacturing appliances of daily use in the home. Philips Hue is an IoT management platform For the connectivity of user's home appliances to their IoT smart space [9]. The dashboard considers usability and best user experience factors allowing connectivity of home appliances to smart home IoT network in a natural way; users can also monitor the appliances remotely from the IoT management platform. This platform only focuses on the realization of IoT; scalability, and management issues are also not considered by this platform [10].

The OPEL software is an IoT platform that supports simple and easy programming. The design of this platform considers features of programming language which can enlarge the IoT ecosystem; features examples are portability, extendibility, and high productivity. The platform should provide a mechanism for secure application development by the use of high-level APIs provided by the platform. Some of the functions provided by these APIs are device management and communication between these devices. This platform also supports multiple IoT applications which means that users can install multiple applications with diverse functions. The platform also supports the concurrent execution of these applications, enabling a user to take advantage of using multiple services on a single device. This software platform also enables the communication of IoT devices with host IoT devices; also the host device will able to control the companion IoT device [11].

Glue.things a project developed on the core concept of device integration and real-time communication utilizing some of the well-established open source technologies. This real-time communication is made possible by using the web sockets' most recent technologies, i.e., CoAP and MQTT. The protocols are utilized on real-time mashups of the network data streams, which is finally deployed in a distributed environment. The proposed system is a mechanism for the composition of data streams from web services [12]. Glue.things provides a collaboration platform with JSON data models, REST APIs and web sockets. Node-RED inspires the mashup interface.

Node-RED from IBM's is a DIY approach based flow programming platform recently developed and got popularity due to its browser-based powerful editor [13]. The goal of the developed was to provide a programming tool for wiring hardware devices by reducing the coding effort and technicality for developers. The node-red tool enables users to wire together hardware devices visualized in graphical nodes form. The graphical nodes represent physical devices, web service, and software platforms. The hardware devices wiring approach used by Node-RED is flow programming which will even enable a new IoT user to participate in the awareness of IoT in other fields. Chris Simpkin et al. [14] proposed mechanism to run workflows like Node-Red on Edge networks, for this purpose they investigated to find out means of migration of Node-RED into a distributed execution environment. The demonstration in this work shows the feasibility of such approaches, traffic congestion detection workflow based on Node-RED is migrated into a decentralized execution environment.

Among the notable contribution in open-source toolkits for creating mainboard is Microsoft .NET Gadgeteer [15], which use the combination of small electronic block approach for mainboard creation. This open-source toolkit enables users to create electronic devices based on their own customization needs; all they need is to combine small electronic blocks on circuitry for the creation of the mainboard. A processor has embedded into the mainboard of Microsoft .NET Gadgeteer. Lucio Tommaso De Paolis et al. used an open-source platform called Thingsboard for sensors data collection and analytics[16]. Thingboard provides mechanisms for data collection, processing the data, providing analytics based on rule engine. The visualization module is embedded with more than 30 widgets to visualize a complete IoT smart space.

Among the notable projects from Kickstarter [17] is SAM [18]. SAM does not need any expertise in the field and can be exploited equally by new users as well as expert developers. The approach used is combining electronics modules blocks, all these modules are wireless, which is a cornerstone for innovation, creation, and designs. The mechanism followed by SAM is using the internet with a combination of hardware and software to a project. The programming language used for SAM toolkit is the python.

Mazzei et al.[19] further investigate how customizability will enable general mass involvement in the creation of such products that can contribute to the IoT ecosystem. In literature, some other studies used the same idea; In the case of Feki et al. research study [20], DIY approach is considered among the most trending approach in the future for the development in the IoT field. Low-cost Systems on Chip (SoC) based IoT development following the DIY approach is proposed by the research study of Scott and Chin [21].

Kefalakis et al. [22] presented an OpenIoT project based on the visual development paradigm. These tools are visual development based Integrated Development Environment (IDE) for the development of IoT application and manage its life cycle. A minimal programming approach is used in the development of IoT applications based on semantic IoT architecture. Users model services into graphs in a node-based user interface, which are then converted into SPARQL queries.

IoT applications development in the form of real-time data streams is provided by an IoT platform called Super Stream Collider (SSC) [23]. This platform will enable everyone from new IoT users to an expert to develop custom IoT applications. SSC web-based interface enables users to develop applications for various IoT scenarios by combining linked streams and data sources to IoT resources. For SSC support editor based SPARQL/CQELS query languages; the approach used is simple drag and drop. The platform exploits cloud infrastructure for massive data acquisition, processing, and spreading of the data.

IoT is now applicable in agriculture, how to bring innovations in agriculture using IoT is studied in a field called digital agriculture. Recently DIY approach based IoT application was developed for digital agriculture. Jayaraman et al. [24] discuss such digital agriculture applications which will make naive users monitor the growth of the crop in a form, suggests and support the user with irrigation decisions. They discuss a user interface called Phenonet, which is a zero-programming based novel DIY architecture. The efficiency of this Open IoT platform is evaluated through Phenonet and other several use cases.

In the examples of services for data feeds collection, Pachube [25] is a notable web-centric service. Pachube store information related to various sensors devices and the data feeds provided by these devices over some time. Pachube work on the idea of "triggers, it is capable of data integration, processing, and visualization. The data arrival from hardware or software resources can be defined as a trigger. The response to such trigger is forwarding of the data to particular URL based on rules defined by Pachube, triggering some other relevant triggers. Pachube increases the creativity and development by provides a medium for sharing the feeds triggered or integrated by one user can be used by another user.

M. S. Khan et al. [26] presents a novel CoAP protocol based on DIY architecture. This toolkit design services based on CoAP proxy. The CoAP proxy Communication with the server device is done through CoAP proxy using socket connection. All the details are handled by improved service composition toolkit. The toolkit visual interface enables a beginner user to compose IoT services. User registers and combines virtual objects for physical devices through the drag and drops approach.

Shabir et al. [27] present a DIY based service designer with a zero-programming experience. Node-RED inspires the concept of the research study, the functionalities provided by remote and CoAP devices can be easily customized. Internet Engineering Task Force (IETF) [28] has standardized CoAP is communication protocols for devices having scarce resources. The CoAP server is implemented using the Intel Edison board. The platform was chosen due to growth in its acceptance day by day in the DIY community all over the world. Visual service designers access the CoAP resources via CoAP proxy shared by the CoAP server. The proxy implementation is based on the Californium [29] framework in a programming language called a Java. For storing the configuration, they use a novel approach based on the repository at the cloud. This cloud-based platform ensures the availability of these configurations everywhere. The HTTP protocol is used to handle communication between real devices and the DIY toolbox, whereas CoAP protocol is used to handle communication between the DIY toolbox and cloud.

In literature, we found some AI-based Intelligent solutions for service composition in the IoT environment. Some AI algorithms are used for virtual object management and naming these objects and automatic service composition. An example of this is A framework for IoT objects based on IoT-IMS platform where the naming and management of objects are done intelligently [30]. An intelligent IoT-based project called IoT.est is an effort to support the exchange of information among distributed and different IoT nodes, by the use of semantic technologies. IoT service is created using a testbed, and several use cases are used for validation purposes. IoT.est also not consider management and scalability issues and only focus on the realization of IoT [31].

Tools which are Platform-As-A-Service (PAAS), and it does not provide ease of use to new users are also discussed in the literature. Some of the platforms discussed by Vestergaard, L.S et al. [32] are Dweet.IO [33], ThingWorx [34] and Particle.IO [35]. Closed source is one of the disadvantages of such platforms. Other well known closed source IoT platforms are Google Cloud IoT [36] and AWS IoT platform [37]. To the best of our knowledge, the paradigm of these tools is either desktop-based or web-based. They say that if we can combine these paradigms and integrate it into IoT applications, then more ideal IoT applications can be developed. We now present a summary of the well-known architectures in Table 1. IoT platforms are summarized based on features, i.e., devices management, integration API's, open-source, support for Edge computing, data analytics and visualization.

Table 1. Summary of the existing work.

IoT Software Platforms	Device Managements	Integration	Edge Computing	Open Source	Types of analytics	Support for Visualization
AWS IoT platform [37]	Yes	REST API	Yes	No	Real-time analytics (Rules Engine, Amazon Kinesis, AWS Lambda)	Yes (AWS IoT Dashboard)
IBM IoT Foundation Device Cloud [13]	Yes	REST API	Yes	No	Real-time analytics (IBM IoT Real-Time Insights)	Yes (data analysis and edge data analysis)
ThingWorx - MDM IoT Platform [34]	Yes	REST API	Yes	No	Predictive analytics (ThingWorx Machine Learning), Real-time analytics (ParStream DB)	Yes (ThingWorx SQUEAL)
Particle.IO [35]	Yes	REST API	Yes	No	Yes (Rules Engine)	Yes (Particle Dashboard)
ThingsBoard [16]	Yes	MQTT API CoAP API HTTP API	No (planning in future)	Yes	Yes (Rules Engine)	Yes (30+ Widgets)
Google Cloud IoT [36]	Yes	Cloud IoT API	Yes	No	Yes(BigQuery)	Yes (Google Data Studio)
Microsoft .NET Gadgeteer [15]	Yes	Gadgeteer SDK	No	Yes	No	Yes

3. Methods

In this section, we discuss the proposed methodology of Task allocation for thermal comfort in smart homes. The proposed methodology consists of several steps. Users use the DIY application to add the services they need. For instance, we add thermal comfort service name and its requirements in detail as a case study for this work. The description is analyzed by the IoT server through natural language processing libraries to split it into microservices. Microservices description is analyzed by the microservice analyzer component for generating input tasks. Physical things are virtualized into virtual objects through virtualization mechanism provided by Virtual Device Manager(VDM), VO-Task Mapping Manager (TMM) mappes relevant tasks to relevant virtual objects. TMM loads tasks and virtual objects to the mapping window and visualizes it through Graphical shapes. User maps a task on a virtual object based on the suggestions; A line is dropped from tasks to virtual objects to establish a connection. A mapping window generates mapping pairs of tasks and virtual objects based on these connections.

Some of these tasks and virtual objects pairs are more important than others and should be executed first than the others. The reordering of these pairs is done by VO-Task Scheduling Manager (TSM). TSM has a scheduler component that receives the mapping pairs from TMM and reorders the mapping pairs through scheduling algorithms of the TSM. Task Allocation Manager(TAM) handles scheduling results and allocate tasks into physical device based on the virtual object information in the scheduling results, i.e., id and URI of the virtual object, which virtualizes a physical device. TAM passes the allocated time, device address and task message to deployer, which finally deploys the task on a physical device at the allocated time. Figure 1 represents the proposed methodology.

Good IoT architecture can help develop a robust and scalable IoT solution. In this paper, we propose a five-layered architecture for the design of IoT applications based on tasks composition and virtual objects mechanisms. The proposed architecture is given in Figure 2. Each Layer has its functionality; the functional goal of all layers is to allocate tasks on virtual objects in an IoT environment efficiently. Physical things Layer composed of physical devices, these devices can be actuators and sensors. Examples of actuators can be categorized to pneumatic actuators, hydraulic actuators, electric actuators, and thermal actuators, fan and LEDs are examples of actuators in IoT smart space. Examples of sensors are a pressure sensor, humidity, proximity sensors, smoke sensors, gas and smoke sensors, IR and temperature sensors.

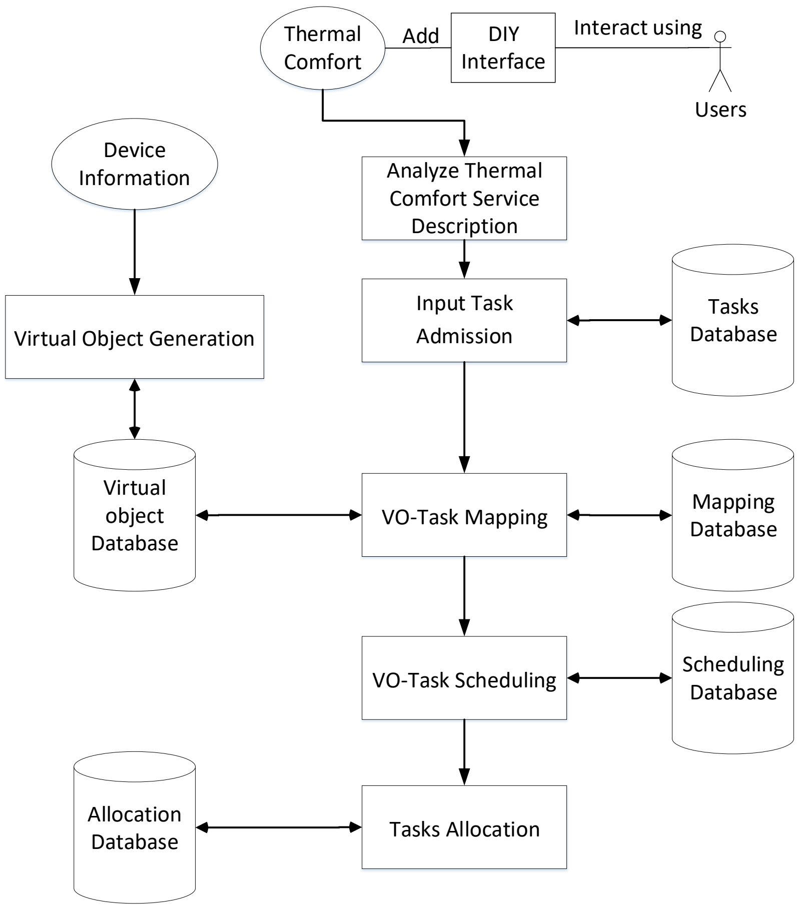


Figure 1. IoT Platform based on Tasks Allocation for Implementing Thermal Comfort System.

The Virtual Objects Layer represents the virtualization of physical layer things; these physical devices are virtualized into virtual objects. VDM manages these virtual objects; virtual objects management includes the registration of physical devices to IoT server registry, Creation of virtual for the physical device and connectivity of physical devices to virtual objects. These virtual objects are visualized through sensors and actuators icons on Google map. Each marker icon represents a virtual object in the interactive map environment; the marker is placed on the latitude and longitude of physical device location on the building or other places where the device is physically deployed. Each virtual object performs specific actions and having certain properties representing attributes of the physical device.

The Micros service layer is responsible for analyzing the description of services and splitting these services into functional units based on the analysis results. A group of virtual objects provides some of these services and hence, this layer performs the role of interconnection between Virtual objects. The microservice composition manager(MSCM) module is responsible for the management of this layer. MSCM is responsible for providing the user interface for services composition, services analysis, and microservice generations. MSCM uses two approaches for Micro Services composition, the first approach compose microservices directly from its associated sensors and actuators. The other approach for microservice composition uses logical objects such as proxy server and fuzzy system between sensors and actuators virtual objects.

The tasks layer is responsible for the composition of tasks from the microservices; this layer manages intuitive and easy to use interface for task composition based on the microservices description. The architecture utilizes the IntelliSense approach to output Intellisense dialogues with tasks and micro services suggestions generated from service at the time of service orchestration.

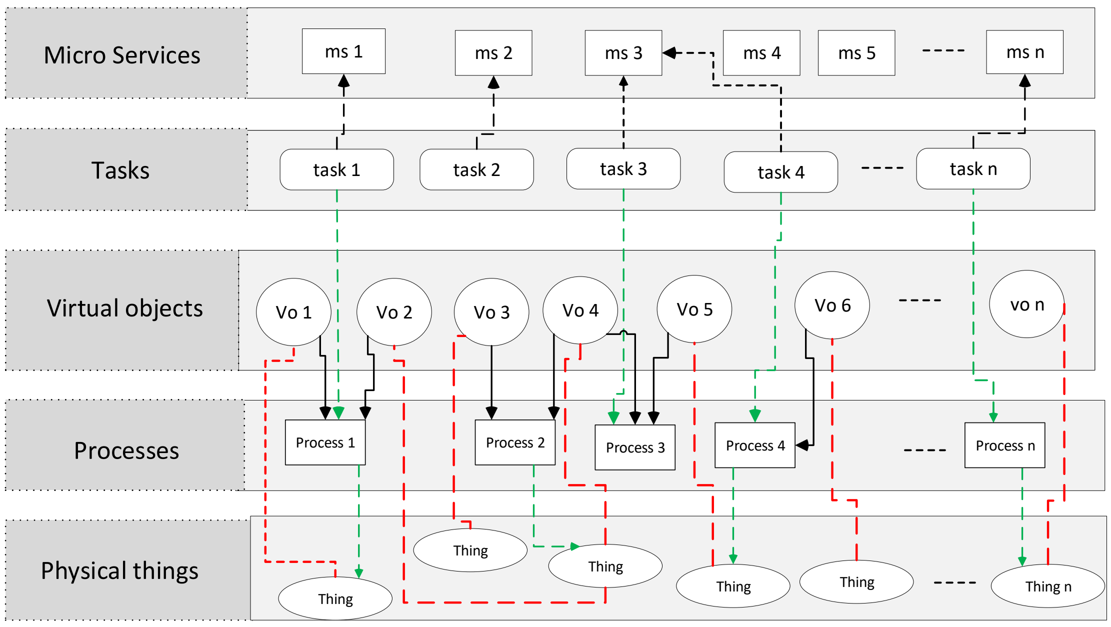


Figure 2. Task Composition Architecture based on Virtual Objects in IoT Environment.

Once tasks are composed in the tasks layers, these tasks are utilized by process layers for further operation on these tasks. The process layer is responsible for the visual environment where tasks can be mapped on virtual objects. Task Mapping is done manually through a drag and drop approach. These mapped task results are utilized by a process created for task scheduling. The scheduling process considers task attributes and scheduling algorithms to generate scheduling results. A process is created for task allocation, which handles scheduling results and allocates tasks to the physical device in the available time slot; finally, tasks are deployed on the physical devices.

4. Design

In this section, we discuss the design of the proposed system developed on the Task Composition Architecture based on Virtual Objects in IoT Environment. We discuss the processes of the platform on top of which thermal comfort mechanism is implemented. The first process is the analysis of thermal comfort servic to generate tasks. Task Generation Manager (TGM) component of the system is responsible for the initialization of TGM components for Services analysis to generate tasks, the design of task generation from microservices is given in Figure 3. Natural Language Processing techniques, i.e., tokenization, stemming, and lemmatization, are used to generate tasks from the description of the microservices.The Part of Speech (POS) tagging technique is used to detect the verb, which helps in deciding the type of the task. Once tasks are produced from the microservice description, the Task manager is also responsible for the interface for displaying the generated tasks profile on the form and filling out the form's visual components.

Task Manager's other responsibility is extracting the configuration profile from both the information generated autonomously through NLP algorithms and from the form information filled by the user. The parsed profile is converted to task data and stored in the task repository. The generated task data is passed to TMM, where it is loaded into a visual interface used for task mapping on virtual objects.

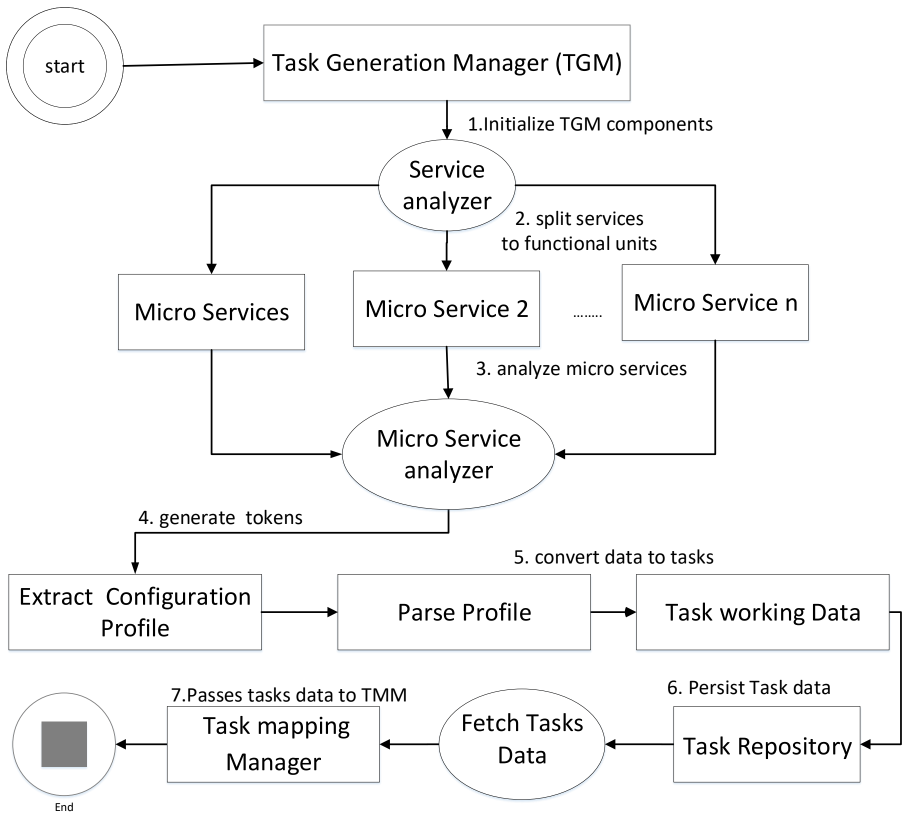


Figure 3. Task Generation from Services.

The second process is the virtualization of physical resources. VDM is the main module responsible for it to form virtual objects (VO). The virtualization process is also visualized through visual components by this module. VDM responsibilities include analyzing the supported protocols on the devices, e.g., CoAP and metadata. VDM also analyzes supported methods and platforms that provide the interface for adding, deletion and updating of the virtual objects. The virtual object is a simulation of the physical device at the virtual IoT smart space. These virtual objects contains all the attributes of the physical device with some extra attributes particular for the virtual IoT environment. These virtual objects also simulate the behaviour of the physical device. Figure 4 represents the design of the Device virtualization mechanism.

The third process is the association of tasks with virtual objects called task mapping in recent studies [8]. Task Mapping Manager(TMM) is the design component responsible for the mapping mechanism. Task Mapping Configuration Flow is shown in the Figure 5. TMM initializes the graphical user interface and libraries needed for the mapping mechanism. The mapping mechanism is drag and drop achieved through mapping libraries. TMM is connected to both task and virtual objects repository. TMM fetches all virtual objects, tasks, and stores it in lists in the main memory. Task and virtual object profiles are extracted and fed into the mapper submodule of TMM. The user uses a drag and drop approach to select a task and map it on the relevant virtual object by dropping a line from the task to a virtual object. TMM visualize the connection between tasks and virtual objects using a graphical line from task to a virtual object. TMM saves the connections between virtual objects and tasks to a database repository.

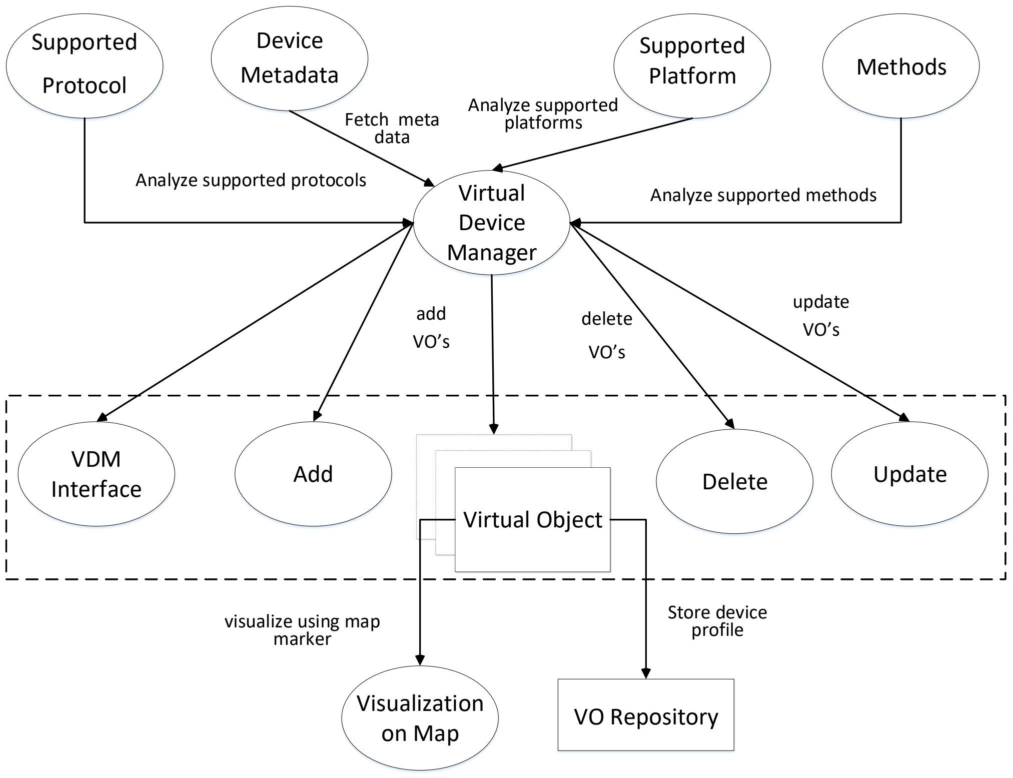


Figure 4. Virtualization of Things.

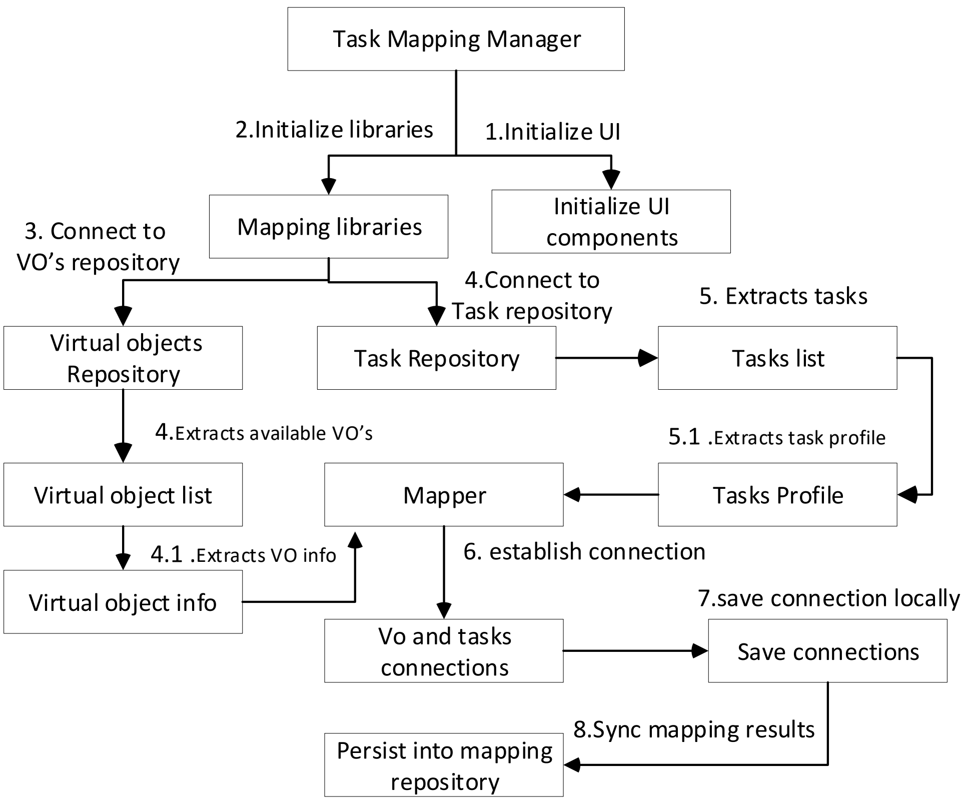


Figure 5. Task mapping Configuration Flow.

The fourth process is the scheduling of the mapped pair in time. Scheduling Manager is responsible for scheduling the Mapping Results on the physical devices. The Scheduling Manager manages the whole flow of the information in the scheduling window. Scheduling Manager is connected to Task Mapping Repository, Virtual Object Repository, and Task Repository.

Scheduling Manager first initializes the Scheduling window graphical user interface. Task mapping results are then fetched from task Mapping repository. Task mapping results contain the ID of a task, and a virtual object, based on these ID's tasks and virtual objects are fetched from their repositories. The design of the scheduling configuration is given in Figure 6. Once tasks and virtual objects are loaded next step is the ordering of the tasks, the primary purpose of task scheduling is a reordering of tasks for task allocation in IoT application.

Scheduler module fetches virtual object information and task profile and creates a scheduling process using the scheduling algorithm. The scheduling process outputs the scheduling order, which is reordering of the mapping pairs. Scheduling Manager saves these results into the scheduling repository in the pairs format task id, virtual object id, scheduled time. Task Allocation Manager utilizes these scheduling results for the allocation of tasks on the physical devices connected to the virtual objects.

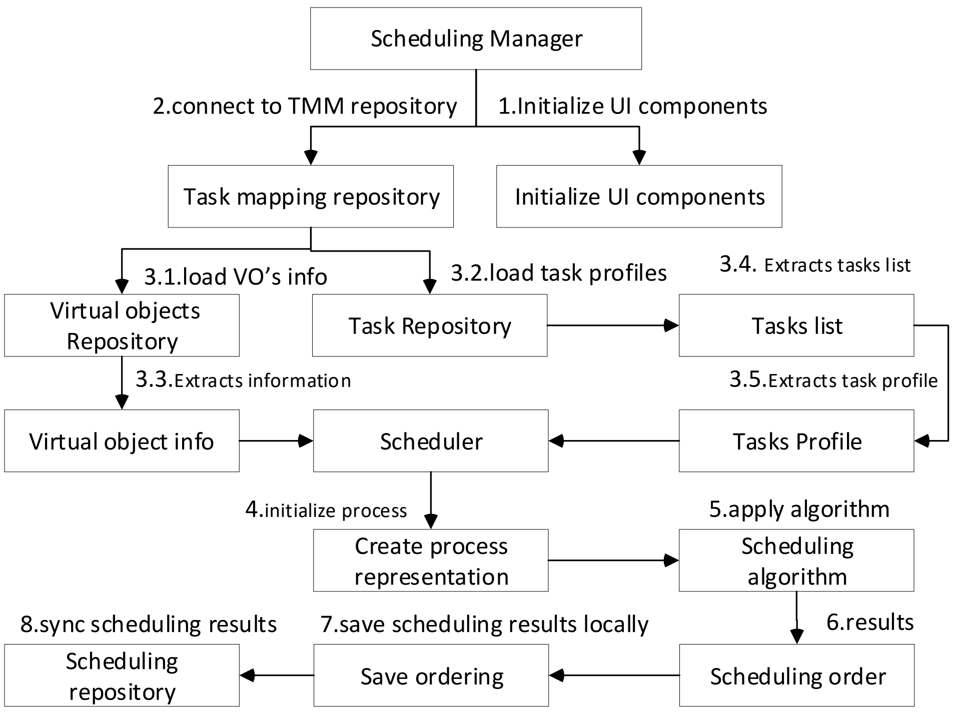


Figure 6. Scheduling configuration Flow.

5. Implementation and Results

In this research, we developed two DIY based applications on the proposed architecture. One application is desktop-based developed in .NET technology; the other is web-based developed in python. Each of the application projects has two main components, one is the IoT server and the other the DIY toolbox, Table 2 explains the IoT server technology stack, whereas Table 3 explains the technology stack of the DIY toolbox. The IoT server is installed on Raspberry PI, all physical resources, i.e., sensors and actuators, are connected with the IoT server. Arduino is used for the support of sensors, i.e., MQ-2 sensor. A registry is maintained of the devices with its current status, the status of the device can be connected, disconnected, available or busy. The implementation at the server side is done through the Python programming language.

Table 2. IoT server's Tech stack.

Component	Description
Operating System	Fedora
Hardware	Raspberry PI, Arduino
Memory	1 GB
Server	Flask Webserver
Resources	sensors and actuators
Programming language	Python
Integrated Development Environment	Vim

Flask, which is a python based framework popular due to the fact that it is widely used in the industry and acceptable to the developers because it is a Model View Controller (MVC) paradigm. For the experimental purpose of this study, different physical resources such as actuators and sensors are used. Examples of actuators used are LEDs and fans, whereas among the sensors used are a hybrid sensor called BME 280, at which three sensors are embedded, i.e., temperature, pressure, and humidity sensors. Other sensors used as physical resources are particulate matter (PM2.5 and PM10) and Gas sensor Co2.

Flask server-side application is responsible for the registration of these physical resources. This server-side application identifies each physical resource by a unique identifier called URI for every resource. To request these remote resources applications such as our DIY toolbox is accessed through these URI by sending an HTTP request to the IoT server, the server give an HTTP response containing JSON object which is then parsed by the DIY Toolbox.

Table 3. DIY based Task and Virtual Objects Composition toolbox Tech stack.

Component	Description
Operating System	Windows 10, 64 bits
CPU	Intel i3-2120 CPU @3.30 GHz(4 CPUs)
Memory	16 Gigabyte
Programming Language	C Sharp, Python
IDE	Visual Studio 2019, Community Version
Libraries	Mind Fusion Diagraming, GMap, MySQL, Ribbon, Json.NET
Persistence	Json files in synch with cloud SQL for MySQL

The toolbox provides the graphical user interface to IoT application users where they can virtualize the physical resources connected to their IT smart space. Based on the services needed by the users, tasks are composed through DIY application and flask server. These tasks are mapped to virtual objects based on the mapping mechanism explained in the design section. The DIY toolbox allocates tasks on physical resources based on task scheduling results. The DIY toolbox provides all these functionalities through simple dragging and dropping approach. In the future, we will focus on developing intelligence to the toolbox for making the IntelliSense and suggestions to users more efficiently. For the ease of the user, we will use natural language processing algorithms such as the one used by the Alexa application.

Figure 7 represents the structure of the DIY applications developed for the evaluation of the architecture client application in C Sharp which executes over the common language runtime, and the web-based DIY application runs on python. The IoT server has handlers for HTTP requests from the DIY toolbox and provides an HTTP response containing JSON objects to the toolbox. The IoT server provides VDM component of the toolbox requests for virtualization of a physical device and devices information. The response for virtualization requests contains device metadata and other attributes of a physical device. Requests containing the analysis of services of the IoT environment are handled by the service configuration manager(SCM) a module developed in python at the IoT server. The requests are parsed by the request parser and responses are generated by the response manger.

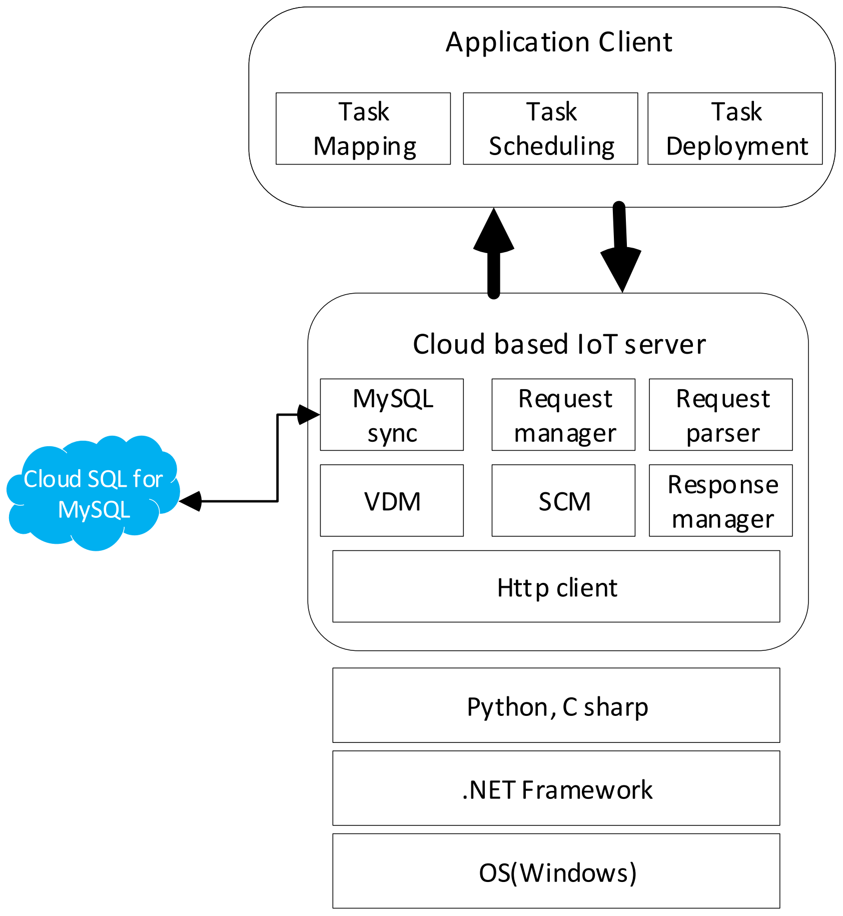
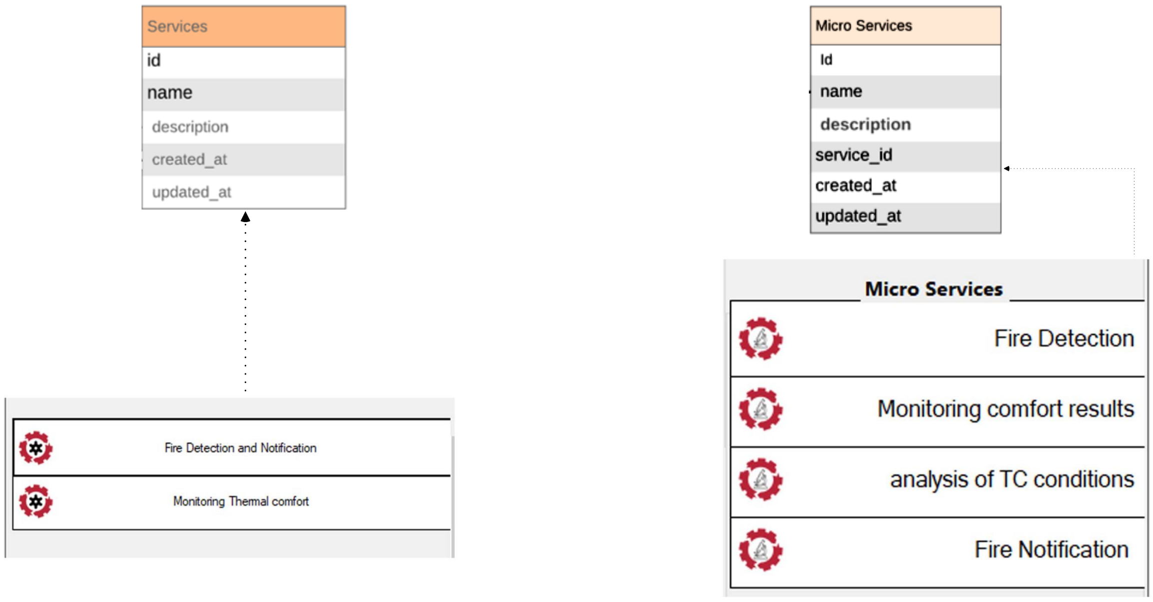


Figure 7. Task and Composition DIY toolbox layered architecture Structure.

5.1. Execution Flow

First, the services added for thermal comfort monitoring is analyzed and microservices are generated. IoT resources are available to the users through IoT services. The user adds services through the DIY application; Microservices are generated autonomously from services description. Natural Language Processing POS technique is used to identify conjunction in the description of the service and the Bigram model is used for Semantic Analysis. Python's NLTK library is used for the tokenization and POS tagging. For the Bigram model python, ngram library is used. Figure 8 represent the implementation results of services and microservices. Section (a) of Figure 8 visualize the services composed manually by the user through the DIY application Interface. Section (b) of Figure 8 shows the microservices results generated based on the description of the service through Natural Language Processing techniques.

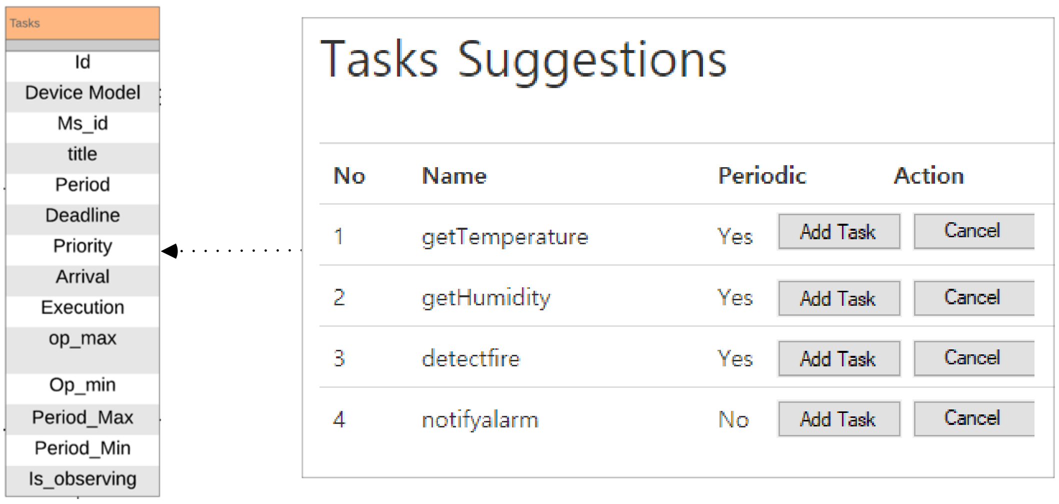
Each service and microservices having a context menu which can be open by right-clicking on the service or microservice. The context menu contains a command for creating new services. Delete command can be used to delete a service or microservice, and View command can be used to display a property window where description and various attributes of services and microservices are visualized through graphical user interface. A service named 'Monitoring Thermal comfort' are used to generate two microservices, namely, Monitoring Thermal comfort results and analysis of Thermal comfort conditions. For better visualization, if the microservice title is long, a short title is generated by abbreviating the verb tokens.



(**a**) Services (**b**) Micro Services

Figure 8. Service and Micro Services Generation.

Once microservices are generated, they are analyzed to generate tasks. Task generation from microservices is visualized in the Figure 9. The figure shows task suggestions based on the microservices description analysis by the DIY Application. Users can either ignore the task suggestions by clicking the cancel command button or utilizing the suggestion by pressing the Add Task command button. The task information is then used by the Task Manager module to fill the window form for adding a task profile. The user fills the rest of the information or modifies the task title, and then saves the task by clicking the add command button. If the user wants to cancel the task profile adding operation, the user has to click the close command button.



Tasks Suggestions

No	Name	Periodic	Action	
1	getTemperature	Yes	Add Task	Cancel
2	getHumidity	Yes	Add Task	Cancel
3	detectfire	Yes	Add Task	Cancel
4	notifyalarm	No	Add Task	Cancel

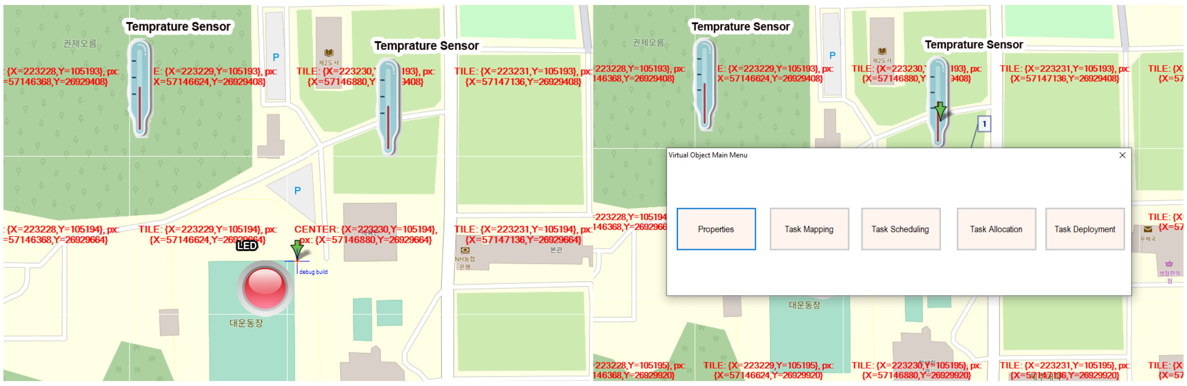
Figure 9. Task Generation from Services.

The task profile is then added to the local database table developed in MySQL in our case, but any database server can be used. The task profile is finally added to the cloud-based MySQL database. Each task is fed to the Task Mapping Manager which is responsible for visualization in the mapping window.

Each task is having a context menu, from which various operations on the task can be performed. Tasks can be added, deleted and previewed in the graphical user interface of the DIY Application.

The next process is device virtualization. As already discussed in the design section, VDM is the main module responsible for the whole virtualization process of physical things into Virtual objects. The device virtualization process is also visualized through an interactive map environment based on the GMAP.NET library. A Virtual object is represented through a map marker using a visual icon of the physical thing. For example, Figure 10a represents the Interactive map with three physical things virtualized. Two temperature sensors are visualized through a temperature icon image. LED Actuator is visualized through Red LED icon image. Each virtual object is having an options window from where properties of the virtual object can be previewed. Figure 10b is showing the options menu which contains commands button for opening windows of task mapping, task scheduling, task allocation, and Task deployment. Each of these windows contains mapping, scheduling, allocation, and deployment modules.

Physical things which are registered and currently connected to IoT server are visualized on the map at a location based on latitude and longitude address of the physical device. The virtual objects configuration is stored in the form of JSON docs temporarily which is then stored to MySQL database at a local server and finally synced to the cloud-based MySQL database.



(**a**) Map with Virtual objects (**b**) Virtual objects option Menu

Figure 10. Device Virtualization and Visualization on Map.

After the virtualization of physical things and Task generation, the next step is the mapping mechanism. The Mapping mechanism from the implementation perspective is presented in Figure 11. The virtualization module extracts all the available virtual objects from the virtual objects repository and feeds the virtual objects to the mapping module. The task manager extracts the candidate tasks from the task repository and feeds them to the mapping module. In the figure, Virtual objects are visualized at the left side of the mapping window as rounded cornered rectangles, whereas tasks are visualized at the right side of the mapping window as circle shapes. A window interface utilizing Mind Fusion diagramming library enables the user to do a simple drag and drop operation of mapping. A line is dragged from the task and dropped at the virtual object establishing a connection between the virtual object and task. Once the connection is made, the Generated Mapping results are saved into it the mapping repository. Mapping repository is a relational table of MySQL database at the local server and cloud MySQL. One mapping connection is stored as one record of the mapping repository table at the MySQL database. The mapper repository stores the id of participating tasks and the id of virtual objects and the mapping time as a single row. In the figure 'Mapped Task' relational table is the repository where the results of the mapped tasks are stored.

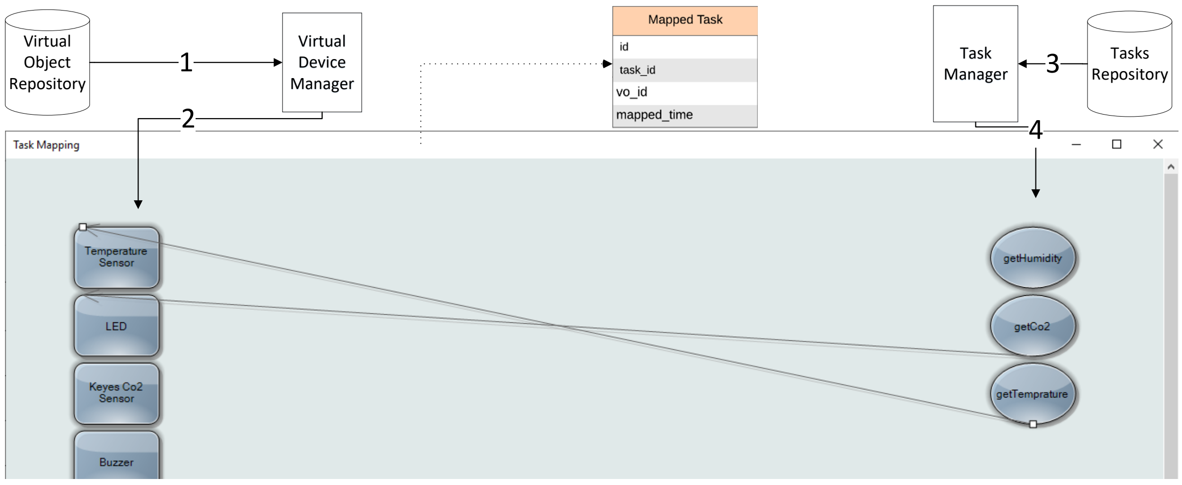


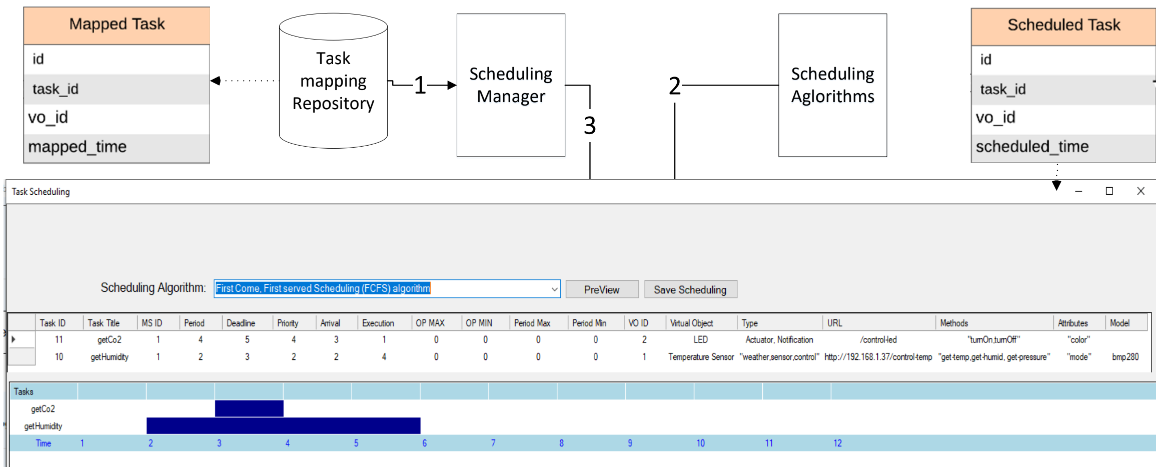
Figure 11. Task mapping from Implementation Perspective.

After task mapping, the next step is task scheduling. As discussed earlier, mapping results are pairs of tasks and virtual objects, i.e., { task id, vo id, mapped time }. In the context of this research study, Task scheduling is referred to the ordering of these mapping pairs to minimize the network latency. So Task scheduling changes the order of mapping pairs such that more important tasks are allocated on the devices first, the importance of tasks is calculated from various task attributes through scheduling algorithms. We used the concept of traditional scheduling algorithms for the ordering of the tasks. The scheduling window is embedded with these algorithms and user select algorithms for the ordering of tasks. The ordered tasks are visualized through table and grant chart. Table 4 lists task attributes considered by the scheduler for reordering the tasks.

Table 4. Task attributes important for task Scheduling.

Component	Description
ID	Identifier of a the task
Period	The period for which the task is assigned to the physical device
VO ID	ID of virtual device to which task is mapped
Deadline	Deadline of the task
Arrival	Arrival of the task
Priority	Priority of the tasks in numbers
Execution	The execution period the task

Figure 12 represents the task scheduling mechanism from an implementation perspective. Task scheduling algorithms are used to produce optimal task orders based on the attributes of mapped tasks. Algorithms we considered for scheduling are First Come, First served (FCFS), Shortest Task Execution First (STEF), Shortest Task deadline Time (STDT), highest Priority Task First(HPTF) and priority-based round-robin. The scheduling manager is responsible for fetching the task and virtual object profiles. It also fetches mapping results of these tasks from the mapping module. The interface provided by scheduling manger is used to choose an algorithm and preview the scheduling results on the table and grant chart. Scheduling algorithms consider an attribute or group of attributes mentioned in Table 3. For example, FCFS considers only the arrival attribute, whereas STDT considers the deadline attribute only. Human inspection on the scheduling results is currently necessary. In the future, we are considering an autonomous intelligent approach for task scheduling. Task scheduling manager saves the generated results into local and cloud-based MySQL repository. The scheduling results are stored in the scheduling repository, which is fed into the task allocation manager for allocation of the tasks on physical devices.



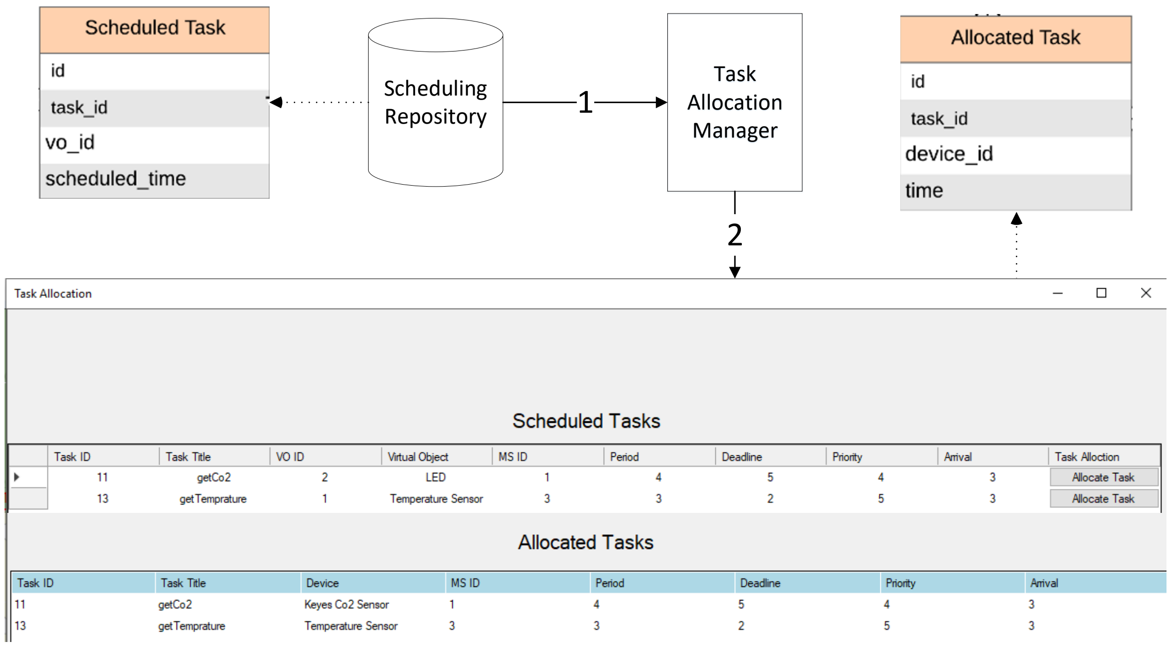
Scheduling Algorithm: First Come, First served Scheduling (FCFS) algorithm

Task ID	Task Title	MS ID	Period	Deadline	Priority	Arrival	Execution	OP MAX	OP MIN	Period Max	Period Min	VO ID	Virtual Object	Type	URL	Methods	Attributes	Model
11	getCo2	1	4	5	4	3	1	0	0	0	0	2	LED	Actuator, Notification	/control-led	"turnOn,turnOff"	"color"	
10	getHumidity	1	2	3	2	2	4	0	0	0	0	1	Temperature Sensor	"weather,sensor,control"	http://192.168.1.37/control-temp	"get-temp,get-humid, get-pressure"	"mode"	bmp280

Figure 12. Task scheduling from the Implementation Perspective.

5.2. *Execution of Tasks Allocation of Thermal Comfort System*

Once all the processes is done by the platform, the actuall deployment happens which is the real running of the service on IoT devices installed in smart homes. The task allocation manager is responsible for the allocation of the task to correct physical devices based on the scheduling pairs. The physical device is accessed through the URI attribute of the virtual object. As discussed earlier, the Raspberry PI based IoT server keeps track of all the devices by maintaining the registry of the connected devices. URI contains a device model and encoded Message, e.g., getTemp. This Message is parsed and decoded by the IoT server to allocate the task on the physical device. Figure 13 shows the implementation of the task allocation mechanism.



Scheduled Tasks

Task ID	Task Title	VO ID	Virtual Object	MS ID	Period	Deadline	Priority	Arrival	Task Allocation
11	getCo2	2	LED	1	4	5	4	3	Allocate Task
13	getTemprature	1	Temperature Sensor	3	3	2	5	3	Allocate Task

Allocated Tasks

Task ID	Task Title	Device	MS ID	Period	Deadline	Priority	Arrival
11	getCo2	Keyes Co2 Sensor	1	4	5	4	3
13	getTemprature	Temperature Sensor	3	3	2	5	3

Figure 13. Task allocation from Implementation Perspective.

The allocation results are stored into the allocation repository, storing the allocation results is helpful for the deployment manager for the deployment of tasks on the physical devices. The repository contains information at the format of task, physical device, allocated time, a background service is managed by the deployment manager to check the allocation repository continuously and deploy the tasks on the physical devices at the allocated time.

5.3. Maintaining Thermal Comfort in Smart Homes: From Service Analysis to Tasks Allocation

In this section, a brief overview of maintaining Thermal Comfort in IoT smart space is presented which summarized the use of the platform and the accomplishment of service goal. The IoT smart space in this case study is the Mobile Computing lab, at Jeju National University. The example case study has been illustrated in the light of the proposed architecture, signifying the strengths of the proposed architecture. Table 5 describes the case study at service, microservice and task levels. Service 1 in the case study for the Thermal Comfort detection and Notification. This service is analyzed from its description to split the service into functional units called microservices.

For Example, there are two microservices with Id's 1.1 and 1.2 belonging to Service ID 1. There are six tasks in Microservice with Id 1.1 and 2 tasks in Microservice with Id 1.2. For this case study, Raspberry PI is used as an IoT server and a PC server for Data Staging between sensors and Cloud SQL based MySQL. Sensors such as MQ-2 Gas is connected to Arduino ESP 8266 board because these sensors belong to analog sensors family and connecting to Raspberry PI would require another hardware such as digital to analog which will add a further overhead and cost on the overall system. The resources we use are shown in the experimental setup shown in Figure 14. From the Experimental setup, it can be understood that The PC Server is responsible for accumulating all the data from Http client and IoT servers. All sensors are either connected to the Http client or the IoT server installed on Raspberry PI. The PC server syncs the data with Cloud SQL for MySQL and deploys tasks on the physical devices either through the HTTP client or through the IoT server.

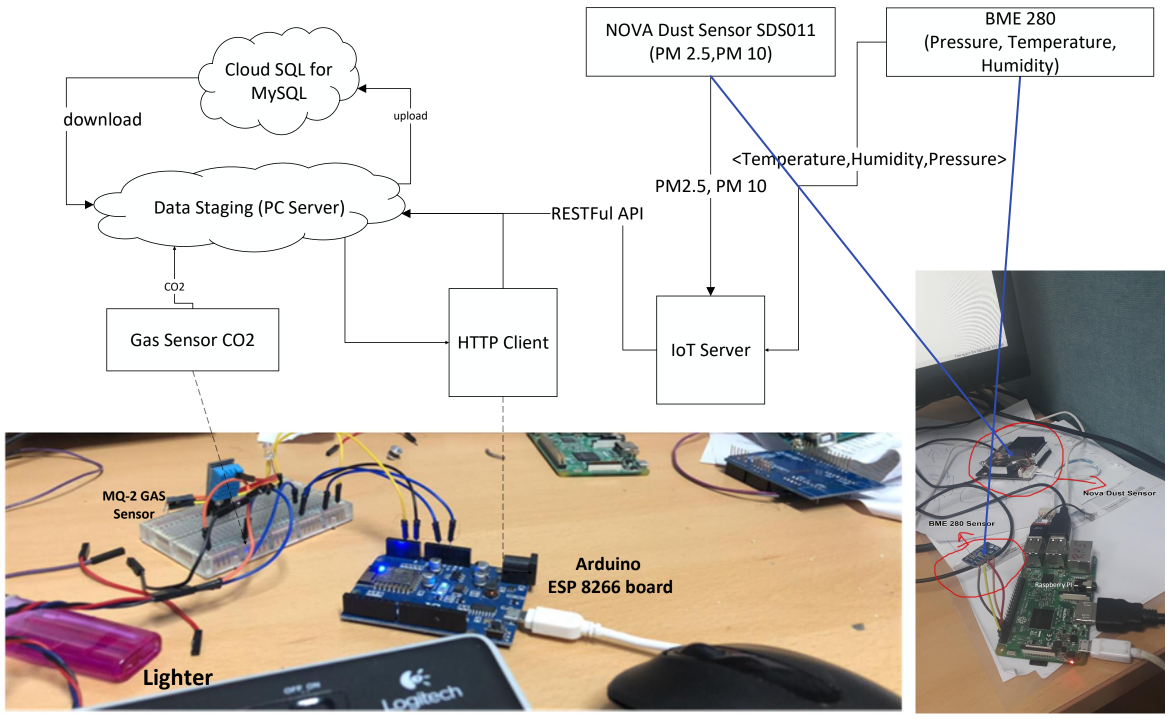


Figure 14. Embedded Hardware for Implementation of Maintaining Thermal Comfort Case Study.

Table 5. Service, Micro service and Tasks Description of Maintaining Thermal Comfort Case Study.

Type	Service ID	Service Name	Service Description				
Service	1	Monitoring Thermal comfort	Monitoring results and analysis of thermal comfort conditions in experimental buildings for different heating systems and ventilation regimes during heating and cooling seasons different heating systems and ventilation regimes				
Service	Service ID	Micro Service ID	Micro Service Name	Host	Edge	Type	Number of Tasks
Micro Service	1	1.1	Monitoring comfort results	PC	Raspberry	Urgent	6
	1	1.2	analysis of thermal comfort conditions	PC	Raspberry	Urgent	2
Tasks Meta Data	Service ID	Micro Service ID	Task ID	Task Description	Type	Task Title	Targeted Resource
Tasks	1	1.1	1.1.1	Get & send temp data	periodic	Get Temp data	BME 280 Sensor
			1.1.2	Receive temp data	Periodic	Receive Temp data	BME 280 Sensor
			1.1.3	Get and send hum data	Periodic	Get hum data	BME 280 Sensor
			1.1.4	Receive Hum data	Periodic	Receive hum data	BME 280 Sensor
			1.1.5	Get and Send CO_2 data	Periodic	Get Wind data	MQ-2 Gas Sensor
			1.1.6	Receive wind data	Periodic	Receive Wind data	MQ-2 Gas Sensor
Tasks		1.2	1.2.1	Detect thermal comfort State based on temp, & hum	periodic	Compute Thermal comfort intensity (TCI)	IOT Server
			1.2.2	turn on fan, or turn off fan	Event-Driven	Predict fan state based on TCI	Actuator (Fan)

6. Performance Analysis

In this section, we present the performance analysis of the proposed architecture. The performance of the proposed architecture is evaluated using a round-trip time (RTT), Response Time, Latency and Task Dropping rate metrics. The RTT in our scenarios is the total time the system takes from task generation till task execution and physical device response back to the DIY application. The DIY application developed based on the proposed architecture maintain logs for each task from its generation until its execution. A log contains information such as task id, target device, deployment status and execution time. For Performance analysis, we calculated the RTT metric by deploying tasks mentioned in Table 5 via the toolbox.

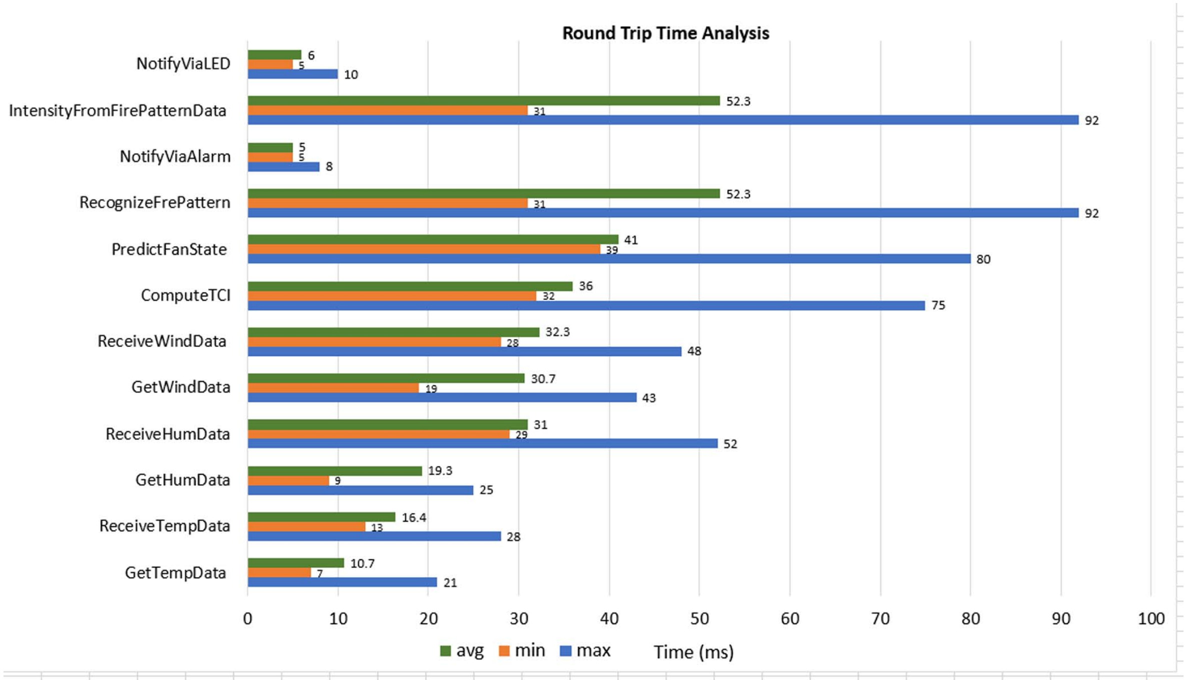


Figure 15. Round Trip Time based Performance Analysis of the proposed Architecture.

Deployment is made based on different task parameters, i.e., execution time, period, arrival time and priority. We deployed these twelve tasks eight times and recorded the total times each task took from task generation to task execution and response back to the application. From 8 observations of each task, we then calculated min, max and Average RTT, which is shown in Figure 15.The RTT is measured in milliseconds, results show that the minimum RTT is 5 ms in the case of the event-driven tasks, for example, task 'Notify Via Alarm' and task 'Notify Via LED'. The maximum RTT is observed for a task is 92 ms which is also minimum and bearable delay for the execution of task which needs input data from other tasks for its complete execution.

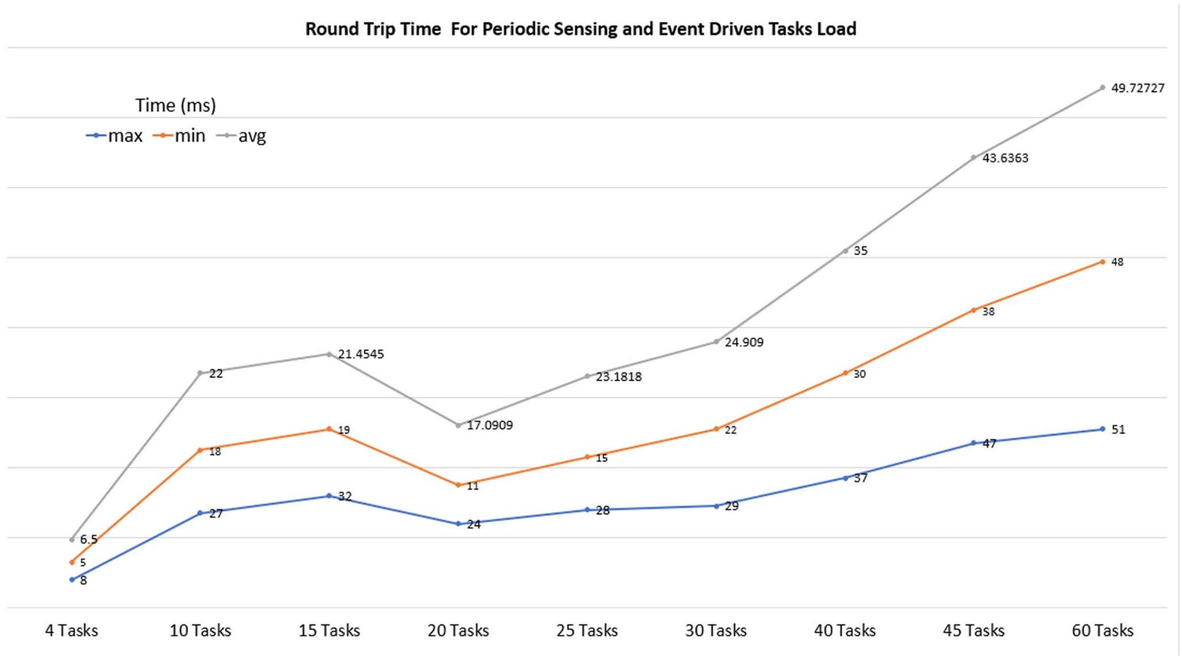


Figure 16. Round Trip Time Analysis for different Task load.

The average RTT for all the tasks executed is 27.75 ms which shows that architecture strength for implementing the architecture in real-time scenarios such as fire safety and road safety. We consider the RTT of Tasks load is the second performance analysis metric. For the RTT of Task load, we simulated 60 tasks in the Apache JMeter load testing tool and analyzed the performance irrespective of the load, RTT of the simulation results are in the order of a few seconds. Figure 16 illustrates the simulation analysis visually through the Line chart. Various tasks load from 4 tasks to 60 tasks were taken for the load testing of the DIY Toolbox. RTT of tasks is directly proportional to the number of tasks, i.e., increasing the number of task loads will increase the RTT, but the average RTT increase is negligible. For instance, for 30 tasks, the minimum RTT observed is 22 ms, the maximum RTT for 30 tasks observed is 29 ms, and the average time for 30 tasks is 24.909 ms. A case study of thermal comfort can be utilized in 12 tasks. The architecture can be utilized for more complex case studies in the IoT environment.

The third metric on which we evaluate the proposed architecture is the response time over time. Figure 17 shows the response time over time analysis of both periodic sensing tasks and event driven tasks.The response time is calculated in Milliseconds for both periodic sensing tasks and event-driven tasks. The maximum and minimum response time for a periodic sensing task is 28.6 ms and 15 ms. Compare to periodic sensing tasks maximum and minimum response time for event-driven tasks are 18.8 ms and 5 ms. As discussed in earlier sections that the sensing tasks sense ambient scenarios and based on contextual data generated, irregular patterns are predicted. Based on these prediction control tasks are activated, which can be executed with a variety of actuators. Having said that, the control tasks, which are event-driven tasks, are of the highest possible priority. The experiments carried out in this section affirms that the proposed architecture always give more priority to event-driven tasks, and this can be witnessed by the lower response time in comparison with sensing tasks.

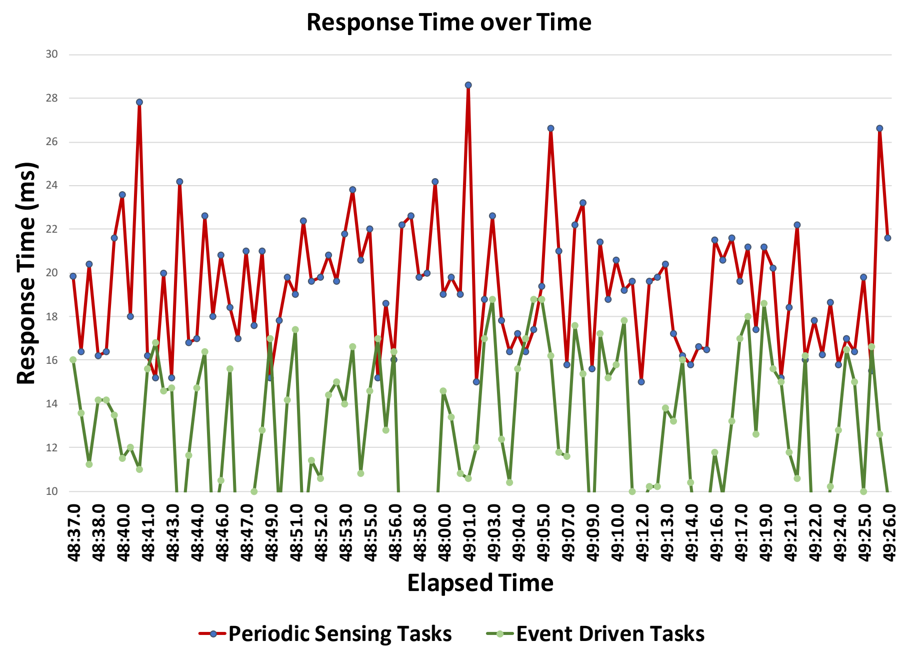


Figure 17. Response Time over Time.

The fourth metric we consider for the performance evaluation is the stability of the architecture in peak load time. Task dropping rate quantitatively accounts for the stability of the system. The lesser number of tasks dropped in peak load time; the more stable will be the system. For this, we carried out the task dropping analysis of the proposed architecture. Figure 18 shows that as the number of input tasks increases, task dropping increases. The minimum task dropping rate is 0 for 26 tasks, and maximum task dropping rate is 14 for 600 tasks. It is evident from the rate of increase, although the number of tasks increases but the rate of the dropping is very steady. Even for 600 tasks, only 27 tasks are dropped, which in other words implies that the performance of the system is 97.7%.

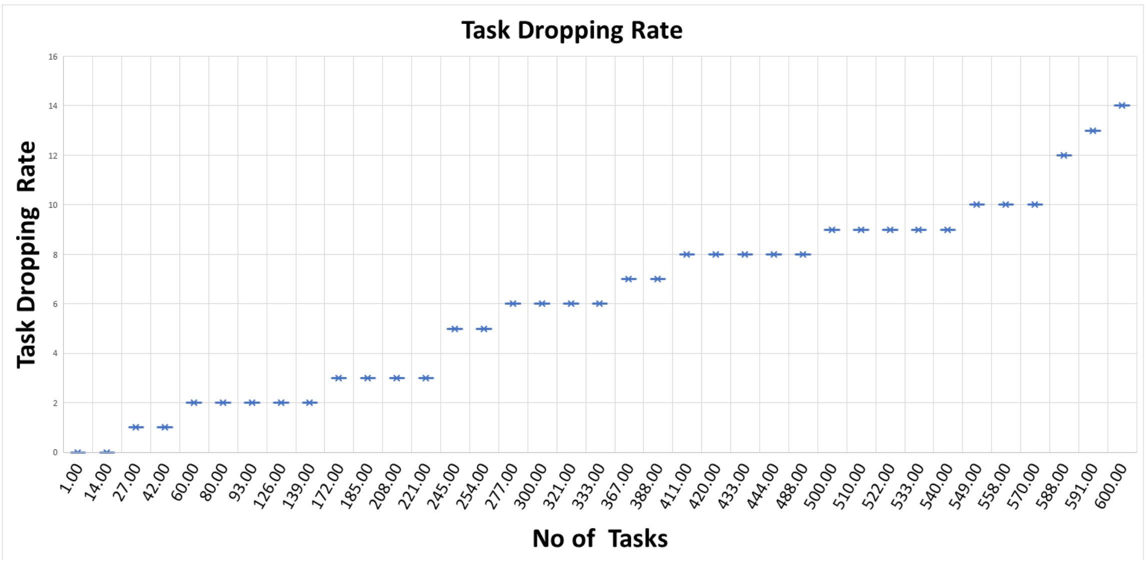


Figure 18. Task Dropping Rate.

Finally, we consider the latency of tasks deployment; for this purpose, we simulated various sets of virtual users in Locust, an open-source load testing tool, for cross-domain requests from our Toolkit. Figure 19 shows the latency of tasks deployment in the simulated environment. During testing, we provided three sets of virtual users, 50 users, 300 users and 1000 users. Minimum latency for 50, 300 and 1000 users are 49 ms,71 ms, and 97 ms, respectively. Maximum latency of 50, 300 and 100 users are 358 ms, 455 ms and 830 ms, respectively. Finally, the average for 50, 300 and 1000 users are 238 ms, 327 ms and 432 ms.

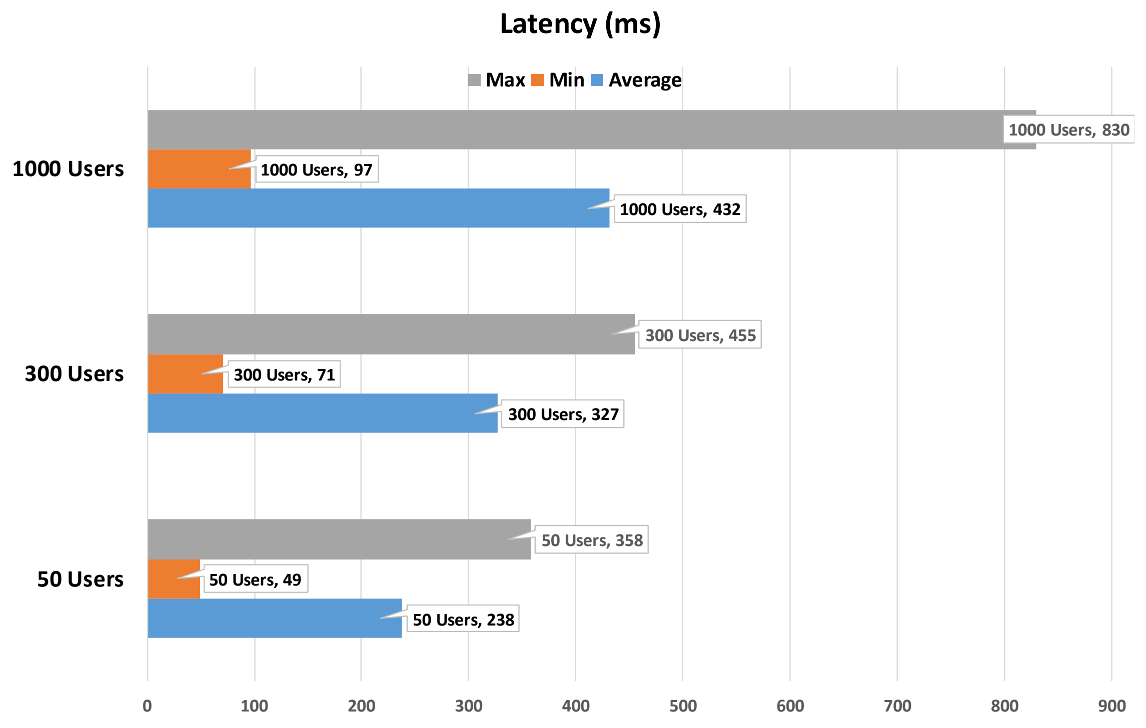


Figure 19. Task Deployment Latency.

Comparison and Significance

To the best of the authors' knowledge, Task Composition Architecture based on Virtual Objects in IoT Environment is the first-ever attempt towards task allocations in the IoT environment using a DIY approach. There are only a few user interfaces for IoT integration platforms such as Node-RED, AWS IoT, NoFlo, ioBroker, CoAP prototype-based DIY interface for service customization of Remote IoT devices. These user interfaces are the only integration platform for the internet of things and do not provide any task level management. First and foremost, The DIY toolkit developed on the proposed architecture is a complete solution addressing various aspects of task allocation in the IoT environment. These aspects include device virtualization, microservices generation from services, task generation from microservices, task mapping to virtual objects, task scheduling and task allocation and deployment on the physical device. Table 6 provides a comparison of the proposed toolkit and the related IoT integration platforms.

For the comparative analysis, we consider various parameters including, open-source, DIY support, Remote Access, Configuration repository, offline access, cloud support, application type, generic entity support, and task level management. One of the goodness of the proposed architecture is the loosely coupled nature of the modules, making it very easy to maintain. The architecture is well-suited for IoT applications that involve multi-devices and multi-tasks. The proposed architecture is flexible, which means that different algorithms for mapping, scheduling and allocation could be introduced without a major revamp to the architecture. The freedom from hardware dependence, task-level management and DIY nature of the integration platform for the internet of things makes our solution a significant, better and ideal solution towards the realization of IoT network.

Table 6. Comparative analysis of proposed Architecture with state of art IoT platforms architectures.

S.No	Name	Open Source	DIY Support	Remote Access	Configuration Repository	Offline Access	Cloud Support	Application Type	Generic Entity Support	Task Level Management	Programming Language
1	Dweet.IO [33]	Yes	No	No	JSON	No	No	Middleware	No	No	Multi language Support
ine											
2	Particle.IO [35]	No	No	Yes	XML	No	Yes	Middleware	Yes	No	Java
ine											
4	Glue.thing [12]	No	Yes	Yes	JSON	No	No	Web	No	No	Node.js
ine											
5	Node-red [13]	Yes	Partial	Yes	JSON	No	No	web	No	No	Node.js
ine											
6	Super Stream Collider [23]	No	Partial	Yes	JSON/ PLSQL	No	No	Web	No	No	JavaScript
ine											
7	SAM [18]	No	Partial	No	XML	Yes	Yes	Desktop	No	No	JavaScript/ Python
ine											
8	Proposed System	Yes	Yes	Yes	JSON	Yes	Yes	Desktop/Web	Yes	Yes	Csharp.NET/ Python

7. Conclusions

In this paper, we proposed a novel approach towards the design and implementation of a task composition toolbox based on virtual objects in the IoT environment. The proposed system allows the user to perform task allocation on both local and remote physical IoT devices. DIY based toolbox is developed based on usability factors, the intuitive user interface and zero coding approaches for effective task allocation and visualization of all the modules processes make it a novel architecture. We have made cloud services available for data storage. The data from the local server is synced and hence ensures the data available to the user remotely. The DIY toolbox also provides real-time visualization of the virtual objects and whole task allocation process at remote IoT devices. In the future, we will add machine learning to automate task mapping to improve the capabilities of the composition toolbox. Task scheduling can be improved by adding complex scheduling algorithms for real-time scenarios, such as a Fair emergency first. The proposed work can also be improved by considering network latencies and device paring for task allocation autonomously. For the ease of the user, we will use natural language processing algorithms such as the one used by the Alexa application.

Author Contributions: Imran conceived the idea for this paper, designed the experiments and wrote the paper; Shabir Ahmed assisted in experimental design. DoHyeun Kim conceived the overall idea of Design and Implementation of Thermal Comfort System based on Tasks Allocation Mechanism, and proof-read the manuscript.

Acknowledgments: This research was supported by the MSIT(Ministry of Science and ICT), Korea, under the ITRC(Information Technology Research Center) support program(IITP-2019-2014-1-00743) supervised by the IITP(Institute for Information & communications Technology Planning & Evaluation).), and this research was supported by Institute for Information & communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT) (No.2018-0-01456, AutoMaTa: Autonomous Management framework based on artificial intelligent Technology for adaptive and disposable IoT). Any correspondence related to this paper should be addressed to Dohyeun Kim.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Vermesan, O. et al; Friess, P.; Guillemin, P.; Gusmeroli, S.; Sundmaeker, H.; Bassi, A.; Jubert, I.S.; Mazura, M.; Harrison, M.; Eisenhauer, M.;. Internet of things strategic research roadmap. In *A Internet of Things-Global390 Technological and Societal Trends*; PW Delft The Netherlands, 2011; Volume 1, pp. 9–52.
2. Friess, P. *Internet of Things-Global Technological and Societal Trends from Smart Environments and Spaces to Green ICT*; River Publishers: PW Delft The Netherlands, 2011.

3. Stern, B et al. *Getting started with Adafruit FLORA: making wearables with an Arduino-compatible electronics platform*; Maker Media, Inc.: Sebastopol, California, USA, 2015
4. Louis, L. Working Principle Of ARDUINO Aand Using It As A Tool For Study And Research . *Int. J. Control. Autom. Commun. Syst. (IJCACS* **2016**, *1*, doi:10.5121/ijcacs.2016.1203 .
5. Tung, D.M.; Toan, N.V.; Lee, J.-G. Exploring the current consumption of an Intel Edison module for IoT applications. In Proceedings of the IEEE International Instrumentation and Measurement Technology Conference (I2MTC), Turin, Italy, 22–25 May 2017.
6. Chaudhar, H. Raspberry Pi Technology: A Review. *Int. J. Innov. Emerg. Res. Eng.* **2015**, *2*, 83–87.
7. Atzori, L.; Iera, A.; Morabito, G. The internet of things: A survey. *Comput. Netw.* **2010**, *54*, 2787–2805.
8. Ahmad, S.; Mehmood, F.; Kim D.H. "A DIY Approach for the Design of Mission-Planning Architecture Using Autonomous Task–Object Mapping and the Deployment Model in Mission-Critical IoT Systems. *Sustainability* **2019**, *13*, 3647.
9. Xia, F.; Yang, L.T.; Wang, L.; Vinel, A. Internet of things. *Int. J. Commun. Syst.* **2012**, *25*, 1101–1102.
10. O'Flynn, C.A lightbulb Worm?. Details of the Philips Hue Smart Lighting Design. In *Black Hat USA*; White Paper, 2016.
11. Lee, H.; Sin, D.; Park, E.; Hwang, I.; Hong, G.; Shin, D. Open software platform for companion IoT devices. In Proceedings of the 2017 IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, NV, USA, 8–10 January 2017.
12. Lee, H.; Sin, D.; Park, E.; Hwang, I.; Hong, G.; Shin, D. Glue. things: A Mashup Platform for wiring the Internet of Things with the Internet of Services. In Proceedings of the 5th International Workshop on Web of Things, Cambridge, MA, USA, 8 October 2014; ACM: New York, NY, USA, 2014.
13. Heath, N. How IBM's Node-RED Is Hacking Together the Internet of Things. Available online: http://www.techrepublic.com/article/node-red/ (accessed on 12 July 2019).
14. Simpkin, C.; Taylor, I.; Harborne, D.; Bent, G.; Preece, A.; Ganti, R. Dynamic Distributed Orchestration of Node-REDIOT Workflows Using a Vector SymbolicArchitecture. In Proceedings of the IEEE ACM Workflows in Support of Large-Scale Science (WORKS), Dallas, TX, USA, 11 November 2018; pp. 52–56.
15. Villar, N.; Scott, J.; Hodges, S.; Hammil, K.; Miller, C. NET gadgeteer: A platform for custom devices. In Proceedings of the International Conference on Pervasive Computing, Newcastle, UK, 18–22 June 2012; Springer: Berlin/Heidelberg, Germany, 2012; pp. 216–233.
16. De Paolis, L.T.; De Luca, V.; Paiano, R. Sensor data collection and analytics with thingsboard and spark streaming. In Proceedings of the 2018 IEEE Workshop on Environmental, Energy, and Structural Monitoring Systems (EESMS), Salerno, Italy, 21–22 June 2018.
17. Kickstarter Project. Available online: https://www.kickstarter.com/ (accessed on 10 July 2019).
18. SAM: The Ultimate Internet Connected Electronics Kit. Available online: https://www.kickstarter.com/projects/1842650056/sam-the-ultimate-internet-connected-electronics-ki (accessed on 12 July 2019).
19. Mazzei, D.; Fantoni, G.; Montelisciani, G.; Baldi, G. Internet of Things for designing smart objects. In Proceedings of the 2014 IEEE World Forum on Internet of Things (WF-IoT), Seoul, Korea, 6–8 March 2014; IEEE: Piscataway, NJ, USA, 2014.
20. Feki, M.A.; Kawsar, F.; Boussard, M.; Trappeniers, L. The internet of things: The next technological revolutions. *Computer* **2013**, *46*, 24–25.
21. Scott, G.; Chin, J. A DIY approach to pervasive computing for the Internet of things: A smart alarm clock. In Proceedings of the 2013 5th Computer Science and Electronic Engineering Conference (CEEC), Colchester, UK, 17–18 September 2013; IEEE: Piscataway, NJ, USA, 2013; pp. 57–60.
22. Kefalakis, N.; Soldatos, J.; Anagnostopoulos, A.; Dimitropoulos, P. A visual paradigm for IoT solutions development. In *Interoperability and Open-Source Solutions for the Internet of Things*; Springer: Berlin, Germany, 2015; pp. 26–45.
23. Quoc, H.N.M.; Quoc, M.; Serrano, M.; Le-phuoc, D.; Hauswirth, M. Super stream collider–linked stream mashups for everyone. In Proceedings of the Semantic Web Challenge Co-Located with ISWC2012, Boston, MA, USA, 11–15 November 2012; pp. 11–15.
24. Jayaraman, P.P.; Palmer, D.; Zaslavsky, A.; Georgakopoulos, D. Do-it-Yourself Digital Agriculture applications with semantically enhanced IoT platform. In Proceedings of the 2015 IEEE Tenth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), Singapore, 7–9 April 2015; IEEE: Piscataway, NJ, USA, 2015; pp. 1–6.

25. Pachube. The Internet of Things Real-Time Web Service and Applications. Available online: http://www.appropedia.org/Pachube (accessed on 12 July 2019).
26. Khan, M.S.; Kim, D. DIY interface for enhanced service customization of remote IoT devices: A CoAP based prototype. *Int. J. Distrib. Sens. Netw.* **2015**, *2015*, doi:10.1155/2015/542319.
27. Ahmad, S.; Hang, L.; Kim D.H. Design and Implementation of Cloud-Centric Configuration Repository for DIY IoT Applications. *Sensors* **2018**, *18*, 474.
28. Shelby, Z.; Hartke, K.; Bormann, C. *The Constrained Application Protocol (CoAP)*; Internet Engineering Task Force (IETF): Fremont, CA, USA, 2014.
29. Kovatsch, M.; Lanter, M.; Shelby, Z. Californium: Scalable cloud services for the internet of things with coap. In Proceedings of the International Conference on the Internet of Things (IOT), Cambridge, MA, USA, 6–8 October 2014.
30. Chang, K.-D.; Chang, C.-Y.; Liao, H.-M.; Chen, J.-L.; Chao, H.-C. A Framework for IoT Objects Management based on Future Internet IoT-IMS Communication Platform. In Proceedings of the 2013 Seventh International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), Taichung, Taiwan, 3–5 July 2013.
31. Gyrard, A.; Bonnet, C.; Boudaoud, K.; Serrano, M.. Assisting IoT Projects and Developers in Designing Interoperable Semantic Web of Things Applications. In Proceedings of the 2015 IEEE International Conference on Data Science and Data Intensive Systems, Sydney, Australia, 11–13 December 2015.
32. Vestergaard, L.S.; Fernandes, J.; Presser, M. Creative coding within the Internet of Things. In Proceedings of the Global Internet of Things Summit (GIoTS), Geneva, Switzerland, 6–9 June 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 1–6.
33. dweet.io. Share Your Thing- Like It Ain'T no Thang. Available online: http://dweet.io/ (accessed on 12 July 2019).
34. Enterprise IoT Solutions and Platform Technology. Available online: https://www.thingworx.com/ (accessed on 12 July 2019).
35. Particle. Connect Your Internet of Things (IoT) Devices. Available online: https://www.particle.io/ (accessed on 12 July 2019).
36. Shin, S.H.; Yoo, W.S. Sensor Gateway Using Arduino via Google Cloud and IEEE 802.15.4e. In Proceedings of the The Thirteenth International Conference on Internet and Web Applications and Services (ICIW 2018), Barcelona, Spain, 22–26 July 2018.
37. Andore, D.B. AWS IOT Platform based Remote Monitoring by using Raspberry Pi. *Int. J. Latest Technol. Eng. Manag. Appl. Sci.* **Oct 2017 Volume vi, pp. 38–42**